Mini-project #2:

Using time series analysis (TSA) for malware detection

# Brief summary

In this mini-project you will use the dataset to generate a time-series representation of data, on which you will use time-series analysis algorithms. Using information derived from the analysis, you will train a classifier, whose goal is to detect malicious files.

This short guide is based on [this](https://www.sciencedirect.com/science/article/pii/S0950705117305336?via%3Dihub) paper, sections 4.1, 4.2.  
Make sure you read and understand the paper.

You will need data relevant to webmail files only, so make sure you use the correct dataset.

# Generating the time-series

Use all the data to build a time-series per file. Each element in the series stores the number of times the file was downloaded, in the respective time window.

Think – is it better to count the number of times a file is downloaded or the number of machines to which it is downloaded to?

Think – what should be the window size? Minutes? Hours? Days?

The data from the last 2 days will be used for testing the model you build based on the first 5 days, so you should perform the feature extraction procedure separately for the 5 days and the last 2 days.

Start by counting the number of distinct machines seen related to each file (think – maybe you want to take into account only the first time the file was seen on the machine?) in each time-frame. Use a scale of hours (per day) and also (in another time-series) a scale of days (using the whole 5 days).

Analyze the resulting time-series (TS) to better understand them. For example, answer questions like: what is the average count per day? What is the standard deviation? What is the difference between the daily and the hourly scale? Which one should be used? Etc.

*Include in your report some TS graphs and statistics from the analysis you perform and describe any insights you gained.*

# Distances computation

Each file is now associated with a TS. For any two given files, the distance between their TS can be calculated using Euclidean distance. Another option is to use [DTW](https://en.wikipedia.org/wiki/Dynamic_time_warping) (dynamic time wrapping) as the distance measure. Calculate the distances between all TS pairs.

Note that the Euclidean distance measure may require some pre-processing (TIP: consider shifting all TS so that the center of mass will be on 0 in the x axe).

Also note that DTW can be expensive. In that case, consider calculating only the distance between each file and **only the malicious files**.

*Include in your report information about the distance measurements you used, any preprocessing you did, its run-time (theoretical and according to measurements you perform) etc.*

Checkpoint

# Distances analysis

Analyze the distances from (e.g.) the following respects:

* 1. Compare the Euclidean to DTW distances. What are the average distances? What is the distribution of the distances?
  2. For each metric, check what is the distances distribution, median, standard deviation, etc.

*Include in your report the graphs and statistics of the analysis you perform, and describe any insights you gained.*

# Feature extraction

Extract 3 types of features for each file:

1. “global” features (size, prevalence, number of source domains etc.)
2. features based on the distances you calculated - one such group of features can be based on the k-NN (k-nearest neighbors) logic: out of the nearest 1/2/5/10/20 files, how many are malicious? Another option is to calculate the average distance (and median) of the closest 1/2/5/10/20 *malicious* files.
3. features based on the TS of the file - span, max/min/median value, number of peaks etc.

Analyze per feature its distribution, comparing clean vs malicious files. Plot them in a manner that will show their differences/similarities (TIP: you may want to exclude extreme values).

*Include in your report the graph and statistics of the analysis you perform, and describe any insights you gained, like what feature seems more helpful, why the distribution is as it is etc.*

Checkpoint

# Machine learning

Note that our dataset contains only malicious labels. As for clean files, use files which are common and prevalent in the data.

1. Use the features extracted to classify files to clean vs malicious. Make sure you use two (or more) types of classifiers and select the most appropriate one.
2. Use cross-validation to check your results and select the model.
3. Make sure to use also a time-split to test your results. Be careful with what files you test yourself on. The files should only appear in the last two days (otherwise they are part of the training data.).

*Include in your report the graph and statistics of the results of the classification. Make sure you compare the models you use. Do the results from the cross-validation align with those of the time-split?*

Checkpoint

# Bonus – adjust DTW for our needs

DTW uses the entire time-series. Consider the case where a file is only seen for a few hours. Its TS can be very similar to a file which is seen for a few days, however, the DTW may miss this resemblance, as it forces each point in each of the TS to be compared with some point in the other TS.

As a concrete example, assume that one TS is 55678876543 and the other is 06786420000. Where the second TS is equal to 0, we are forced to compare it to (at least) 3, while it is possible that the first TS has a slower decay, and it would have gone all the way down to 0 as well if we have used a wider window. This issue is not a problem with DTW, but with its usage in our domain, where files can be seen for various sizes of windows, and the value ‘0’, used when there are no reports, can augment the results of the DTW.

Think what can be done to overcome this issue. You can either change the DTW calculation or perform some preprocessing to the TS.